DUPLICATE COPY

U.S. DEPARTMENT OF COMMERCE
NATIONAL OCEANIC AND ATMOSPHERIC ADMINISTRATION
NATIONAL WEATHER SERVICE
NATIONAL METEOROLOGICAL CENTER

OFFICE NOTE 346

NUMERICAL SOLUTION OF THE PRIMITIVE EQUATIONS ON THE CONNECTION MACHINE

James J. Tuccillo
Automation Division

December 29, 1988

NUMERICAL SOLUTION OF THE PRIMITIVE EQUATIONS ON THE CONNECTION MACHINE

James J. Tuccillo
National Meteorological Center
Washington, D.C.

1. INTRODUCTION

Since the late 1960's, the hydrostatic primitive equations have been the basis for operational Numerical Weather Prediction (NWP). These equations describe the time rate of change of the three dimensional atmospheric state variables: wind, temperature, moisture and pressure. Numerical time integration, from a set of initial conditions, for a period of 2 to 10 days provides guidance that has become indispensable to the operational forecaster. These numerical solutions require significant computer resources and operational weather centers have sought out the most advanced digital computers available. The most advanced systems, however, are often saturated shortly after installation as the NWP models increase in resolution and sophistication. The nature of the problem is such that a doubling of the spacial resolution, in 3 dimensions, increases the CPU requirements by a factor of 16 and the memory requirements by a factor of 8. The demand for increased memory and computational speed will most likely continue into the foreseeable future as modelers strive for increased accuracy through better spacial resolution and greater sophistication in the representation of physical processes.

The last 20 years has seen remarkable growth in the computational speed of computers and the size of random access memories. (see Fig. 1). The development of vector processors capable of processing many operands in a pipelined manner has been a major development. The CRAY-1 and CDC CYBER 205 are the most popular examples of this architecture. As pipelines machines approach a ceiling in performance do to a limit on the speed of signal propagation in semiconductor

chips, the emphasis of the supercomputer industry has shifted towards the development of parallel architectures. Many computationally intensive tasks are inherently parallel and architectures which can exploit that parallelism can be used to solve problems once thought to be intractable.

Parallel processors have developed along at least three paths. The first involves the interconnection of a few very fast processors to a shared memory. Examples of this approach include the current architectures of Cray Research and ETA/CDC. The second path is represented by the so-called Dataflow machines. In this architecture a detailed analysis of the program is performed. Instructions are generated to perform operations as operands become available thus avoiding the von Neumann bottleneck. The Very Long Instruction Word (VLIW) architecture of Multiflow Computer is an example. The third path involves the interconnection of thousands of relatively slow processors. The Connection Machine (CM) of Thinking Machines Corporation is an example of this type of architecture.

This paper will be concerned with discussing the NWP problem on the CM. Section 2 will present the formulation of an NWP model including the finite difference operators and the organization of calculations. In section 3 the CM will be introduced and the hardware characteristics discussed. Section 4 will describe how the NWP model was implemented on the CM. Section 5 will present performance figures for the model on the CM and several other architectures. Section 6 will be the summary and conclusions.

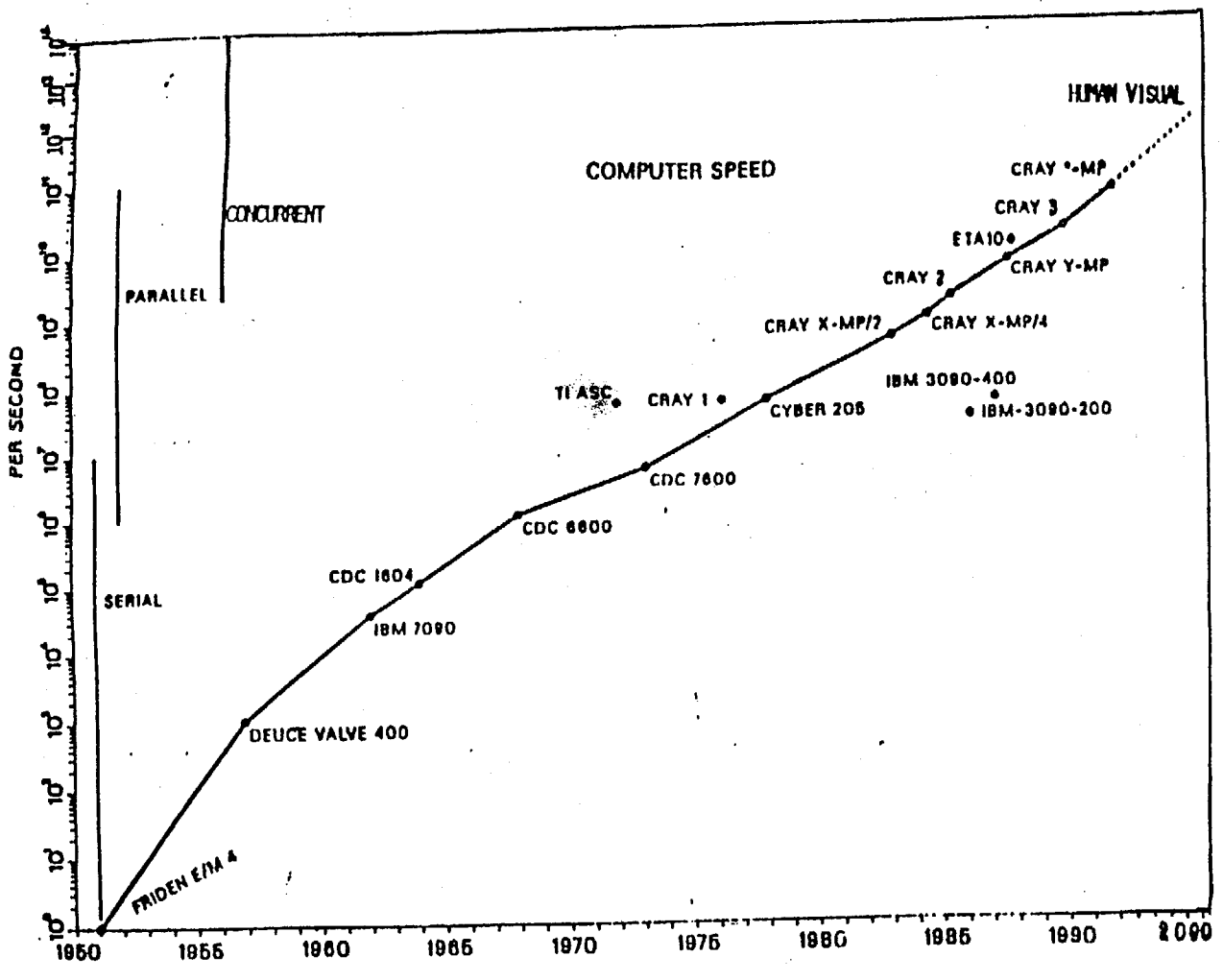

Fig. 1 History of peak supercomputer performance in Megaflops.

2. MODEL FORMULATION

In this section a grid-point, primitive equation model will be presented.

2.1 Primitive equations on a polar stereographic projection

The primitive equations on a polar stereographic projection in the sigma coordinate (Phillips, 1957) are presented below.

$$\frac{\partial u}{\partial t} = -mu\frac{\partial u}{\partial x} - mv\frac{\partial u}{\partial y} - \dot{\sigma}\frac{\partial u}{\partial \sigma} - mC_p\theta_v\frac{\partial P}{\partial x} - m\frac{\partial \Phi}{\partial x} + fv + \frac{uy - vx}{2a^2} + \left(\frac{\partial u}{\partial t}\right)_{turbulence} \quad (1)$$

$$\frac{\partial v}{\partial t} = -mu\frac{\partial v}{\partial x} - mv\frac{\partial v}{\partial y} - \dot{\sigma}\frac{\partial v}{\partial \sigma} - mC_p\theta_v\frac{\partial P}{\partial y} - m\frac{\partial \Phi}{\partial y} - fu - \frac{uy - vx}{2a^2} + \left(\frac{\partial v}{\partial t}\right)_{turbulence} \quad (2)$$

$$\frac{\partial \theta}{\partial t} = -mu\frac{\partial \theta}{\partial x} - mv\frac{\partial \theta}{\partial y} - \dot{\sigma}\frac{\partial \theta}{\partial \sigma} + \left(\frac{\partial \theta}{\partial t}\right)_{turbulence} + \left(\frac{\partial \theta}{\partial t}\right)_{precipitation} + \left(\frac{\partial \theta}{\partial t}\right)_{radiation} \quad (3)$$

$$\frac{\partial q}{\partial t} = -mu\frac{\partial q}{\partial x} - mv\frac{\partial q}{\partial y} - \dot{\sigma}\frac{\partial q}{\partial \sigma} + \left(\frac{\partial q}{\partial t}\right)_{turbulence} + \left(\frac{\partial q}{\partial t}\right)_{precipitation} \quad (4)$$

$$\frac{\partial p_*}{\partial t} = -m^2\int_0^1 \frac{\partial}{\partial x}\left(\frac{u p_*}{m}\right) + \frac{\partial}{\partial y}\left(\frac{v p_*}{m}\right) d\sigma \quad (5)$$

$$\frac{\partial \Phi}{\partial P} = -C_p\theta_v \quad (6)$$

$$p_*\frac{\partial \dot{\sigma}}{\partial \sigma} = -m^2\left(\frac{\partial}{\partial x}\left(\frac{u p_*}{m}\right) + \frac{\partial}{\partial y}\left(\frac{v p_*}{m}\right)\right) - \frac{\partial p_*}{\partial t} \quad (7)$$

where

$$\sigma = \frac{p}{p_*}$$

$$P = \left(\frac{p}{1000}\right)^{0.286}$$

$$m = \frac{2}{1 + \sin\phi}$$

$$x = \frac{2a\cos\phi\cos\lambda}{1 + \sin\phi}$$

$$y = \frac{2a\cos\phi\sin\lambda}{1 + \sin\phi}$$

$$f = 2\Omega\sin\phi$$

and the other symbols have the usual meteorological meaning.

These equations describe the time rate of change of four quantities, u, v, θ, q, which vary in 3-dimensions and one quantity, p*, which varies in 2-dimensions. In terms of Fig. 2, u, v, θ, q, are defined over NX, NY, and NZ while p* is defined over NX and NY.

2.2 Horizontal and Vertical Grid Structure

The horizontal grid is the "B" grid described by Arakawa (1972) and illustrated in Fig. 3. This grid has very good geostrophic adjustment properties and the placement of the variables facilitates the programming effort. The vertical structure is presented in Fig. 4 and represents a standard configuration found in many models.

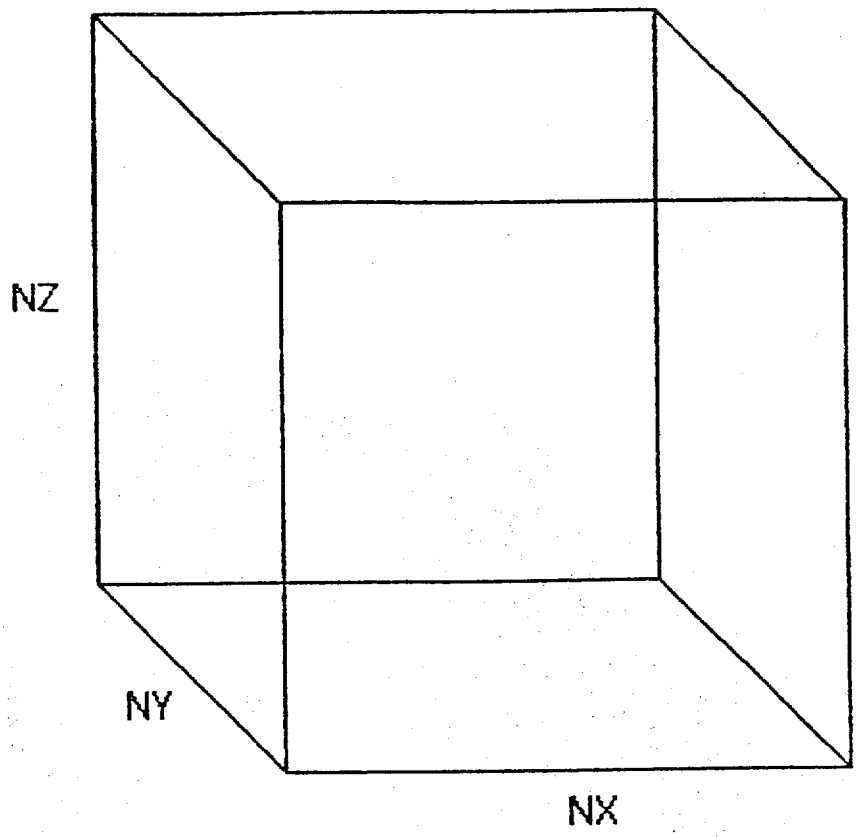

Fig. 2 Geometric definition of NX, NY and NZ.

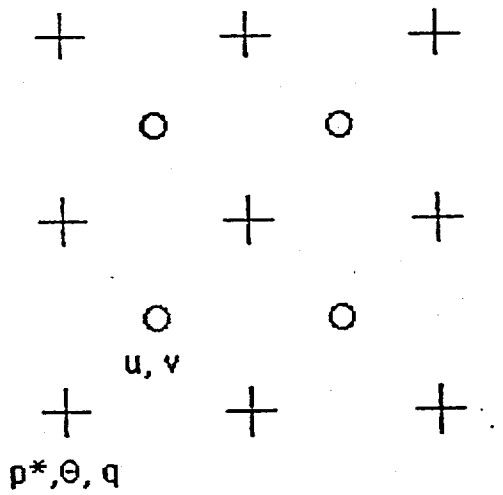

Fig. 3 The Arakawa "B" grid. The circle points represent the location of the u and v wind components and the plus points represent the location of p*, theta and specific humidity.

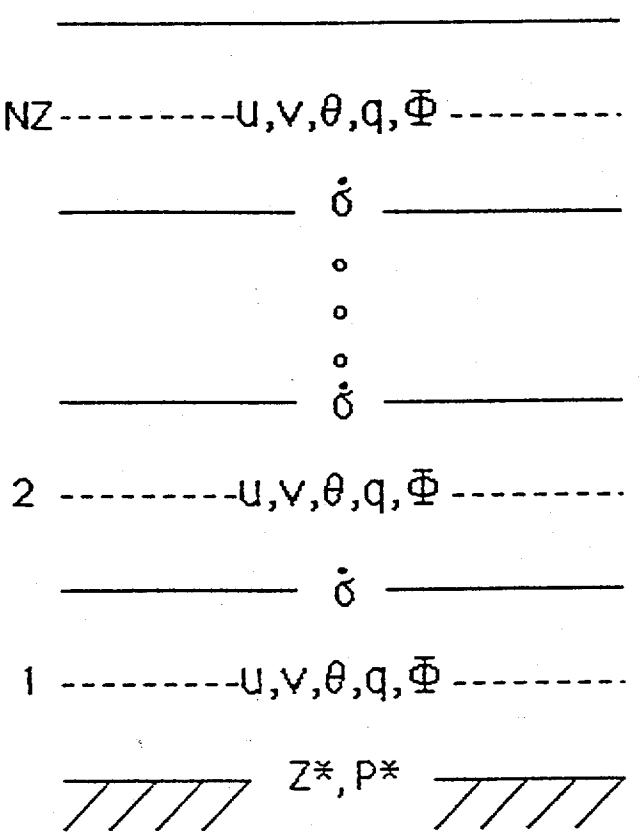

Fig. 4 Vertical sigma structure of the model. The solid lines represent the interfaces between sigma layers and the dashed lines represent the mid-points of the sigma layers.

2.3 Space differencing and averaging operators

The x and y horizontal derivative are approximated by the following formulas:

$$\frac{\partial ( )}{\partial x} \approx \overline{( )_x}^{y} \equiv \left(( )_{i+1,j} - ( )_{i,j} + ( )_{i+1,j+1} - ( )_{i,j+1} \right) * \left(\frac{0.5}{delx} \right)$$

$$\frac{\partial ( )}{\partial y} \approx \overline{( )_y}^{x} \equiv \left(( )_{i,j} - ( )_{i,j+1} + ( )_{i+1,j} - ( )_{i+1,j+1} \right) * \left(\frac{0.5}{dely} \right)$$

where the i and j indices refer to either the plus points for a derivative at the circle points or the circle points for a derivative at the plus points. Please note that i increases in the eastward direction and j increases in the southward direction.

The vertical advection terms are approximated by the following formula:

$$\dot{\sigma}\frac{\partial()}{\partial\sigma} \approx \left(\dot{\sigma}_{k+\frac{1}{2}} \left(\frac{()_{k+1}-()_k}{\sigma_{k+1}-\sigma_k} \right) + \dot{\sigma}_{k-\frac{1}{2}} \left(\frac{()_k-()_{k-1}}{\sigma_k-\sigma_{k+1}} \right) \right) * 0.5$$

where the k + 1/2 and k - 1/2 indices refer to the sigma layer interfaces and the k + 1 and k - 1 indices refer to the sigma layers.

The horizontal averaging operator is as follows:

$$\overline{()}^{xy} = \left(()_{i,j} + ()_{i+1,j} + ()_{i,j+1} + ()_{i+1,j+1} \right) * 0.25$$

where the i and j indices refer to plus points for an average at the circle points and the circle points for an average at the plus points.

The formulation for the vertical integration of the vertical velocity and geopotential are standard and not presented here.

2.4 Time differencing

The time differencing scheme is the split-explicit method described by Gadd (1978). In this scheme the terms responsible for gravity-inertia oscillations are time integrated seperately from the terms associated with advection and physical processes. The advantage of this approach is one of economy since the terms associated with fast moving and meteorologically unimportant waves can be solved with a small timestep for computational stability while the remining terms are solved less frequently with a longer time step. The net result is a considerable savings in computer time over a scheme which solves all terms with a timestep needed for stability of the gravity-inertia waves. Details of the method and a stability analysis can be found in Gadd's paper and the references within.

2.4.1 Gravity-Inertia terms

The gravity-inertia terms are integrated with the forward-backward scheme as described by Gadd. The first step is a forward time difference of the surface pressure tendency equation.

$$p_*^{n+1} = p_*^n + F^n * \Delta t_{gw}$$

where F represents the RHS of (5).

Next the thermodynamic equation and moisture conservation equations are forward time differenced with only the vertical advection terms after the vertical velocity is computed with (7).

$$\theta^{n+1} = \theta^n + VA^n * \Delta t_{gw}$$

$$q^{n+1} = q^n + VA^n * \Delta t_{gw}$$

where VA represents the vertical advection terms of (3) and (4).

With the updated surface pressure, theta, and specific humidity, the geopotential is vertically integrated using (6). The equations of motion, (1) and (2), are then updated.

$$u^{n+1} = u^n + \left[VA^n + PGF^{n+1} + MCT^n + f\left(\frac{v^n + v^{n+1}}{2}\right)\right] * \Delta t_{gw}$$

$$v^{n+1} = v^n + \left[VA^n + PGF^{n+1} + MCT^n - f\left(\frac{u^n + u^{n+1}}{2}\right)\right] * \Delta t_{gw}$$

where VA represents the vertical advection, PGF represents both pressure gradient terms, MCT represents the terms with map coordinates and f is the coriolis parameter.

After two consecutive timesteps, the advective terms of (1), (2), (3), and (4) are evaluated. This procedure is described in section 2.4.2.

2.4.2 Horizontal advection terms

The horizontal advection terms are solved with the Lax-Wendroff scheme (Lax and Wendroff, 1960). In this scheme, provisional values of the prognostic variables are computed at time level n + 1/2. Using these provisional values, the forcing is recomputed and the updated values of the prognostic variables are obtained at time level n + 1. This procedure is represented below.

$$u^{n+\frac{1}{2}} = \overline{\left(u^{n}\right)}^{xy} + HA^{n} * \frac{\Delta t_{adv}}{2}$$

$$v^{n+\frac{1}{2}} = \overline{\left(v^{n}\right)}^{xy} + HA^{n} * \frac{\Delta t_{adv}}{2}$$

$$\theta^{n+\frac{1}{2}} = \overline{\left(\theta^{n}\right)}^{xy} + HA^{n} * \frac{\Delta t_{adv}}{2}$$

$$q^{n+\frac{1}{2}} = \overline{\left(q^{n}\right)}^{xy} + HA^{n} * \frac{\Delta t_{adv}}{2}$$

$$u^{n+1} = u^{n} + HA^{n+\frac{1}{2}} * \Delta t_{adv}$$

$$v^{n+1} = v^{n} + HA^{n+\frac{1}{2}} * \Delta t_{adv}$$

$$\theta^{n+1} = \theta^{n} + HA^{n+\frac{1}{2}} * \Delta t_{adv}$$

$$q^{n+1} = q^{n} + HA^{n+\frac{1}{2}} * \Delta t_{adv}$$

where HA represents the horizontal advection terms of (1), (2), (3), and (4). The n + 1/2 provisional values are computed at the circle points for those prognostic variables defined at the plus points. Similarly, the n + 1/2 provisional values are computed at the plus points for those prognostic variables defined at the circle points.

2.5 Horizontal Boundary Conditions

For this study the model was configured with cyclic boundary conditions in the x-direction and fixed boundary conditions in the y-direction for all prognostic variables. In real data applications tendencies from a larger scale model can be applied to the boundaries in a manner similar to many operational limited-area models.

2.6 Sequence of calculations

For each full timestep there are two gravity-inertia wave timesteps and one advective timestep. The sequence of steps to solve the equations for each full timestep are presented in Fig. 5. For each step in the sequence the calculations take place at all NX x NY horizontal grid points in parallel.

The calculation of the vertical velocity and the geopotential couple each vertical layer to the one below it for the solution of the inertia-gravity wave terms. The horizontal advective terms computed in the second 'k' loop are not vertically coupled. The vertical coupling of the layers is of no consequence for the CM as the parallelism is across the horizontal domain. As we will see later, parallel processing of such a fine-grained algorithm is a problem on the CRAY Y-MP. The CRAY will function best if the code is setup so that multitasking is by vertical layer. The recursive propery of the vertical velocity and geopotential make a restructuring of the code necessary for optimum execution on the CRAY.

GRAVITY-INERTIA WAVE TERMS

FOR GW = 1 TO 2 DO

$p*(n+1) = p*(n) + \ldots$

FOR K = 1 TO NZ DO

$\sigma(K) = \sigma(K-1) + \ldots$

$\theta(K)^{(n+1)} = \theta(K)^{(n)} + \ldots$

$q(K)^{(n+1)} = q(K)^{(n)} + \ldots$

$\Phi(K) = \Phi(K-1) + \ldots$

$u(K)^{(n+1)} = u(K)^{(n)} + \ldots$

$v(K)^{(n+1)} = v(K)^{(n)} + \ldots$

END FOR

END FOR

ADVECTIVE TERMS

FOR K = 1 TO NZ

$u(K)^{(n+1)} = u(K)^{(n)} + \ldots$

$v(K)^{(n+1)} = v(K)^{(n)} + \ldots$

$\theta(K)^{(n+1)} = \theta(K)^{(n)} + \ldots$

$q(K)^{(n+1)} = q(K)^{(n)} + \ldots$

END FOR

Fig. 5 Sequence of calculations to advance the solution by on full timestep. 'K' refers to the vertical layer and 'n' indicates the time level.

3. THE CONNECTION MACHINE

The Connection Machine (Hillis. 1986) is a single instruction/multiple data (SIMD) parallel computer with up to 65536 processors controlled by a conventional front-end computer (see Fig. 6). Each processor has 8K bytes of memory yielding a total memory capacity of 512 Megabytes or 128 million 32-bit floating point numbers. Problems requiring more physical processors than are available are supported through a virtual processor mechanism which is invisible to the user. Each physical processor can simulate several virtual processors with an associated decrease in memory and increase in execution time. For example, if each physical processor simulates two virtual processors then the execution time will double and the memory available for each virtual processor will be 4K bytes. Fig. 7 presents various virtual processor configurations and the associated memory.

Interprocessor communication is handled by a network built on a 12-dimensional hypercube. This hardware supports two mechanisms for communication. The router is the more general mechanism and allows for data to be sent from any processor directly to any other processor. The less general method is refered to as NEWS communication after the four directions on a two-dimensional grid: North, East, West and South. The NEWS mechanism allows for the efficient exchange of information between adjacent processors on a grid.

4. IMPLEMENTATION ON THE CONNECTION MACHINE

4.1 Data structure

The most straight forward implementation of the model on the Connection Machine consist of assigning a virtual processor to each column. The following C* code will define a data structure called "state" which specifies the memory layout within each processor and then creates a variable called "points" which consists of NX by NY instances of the structure.

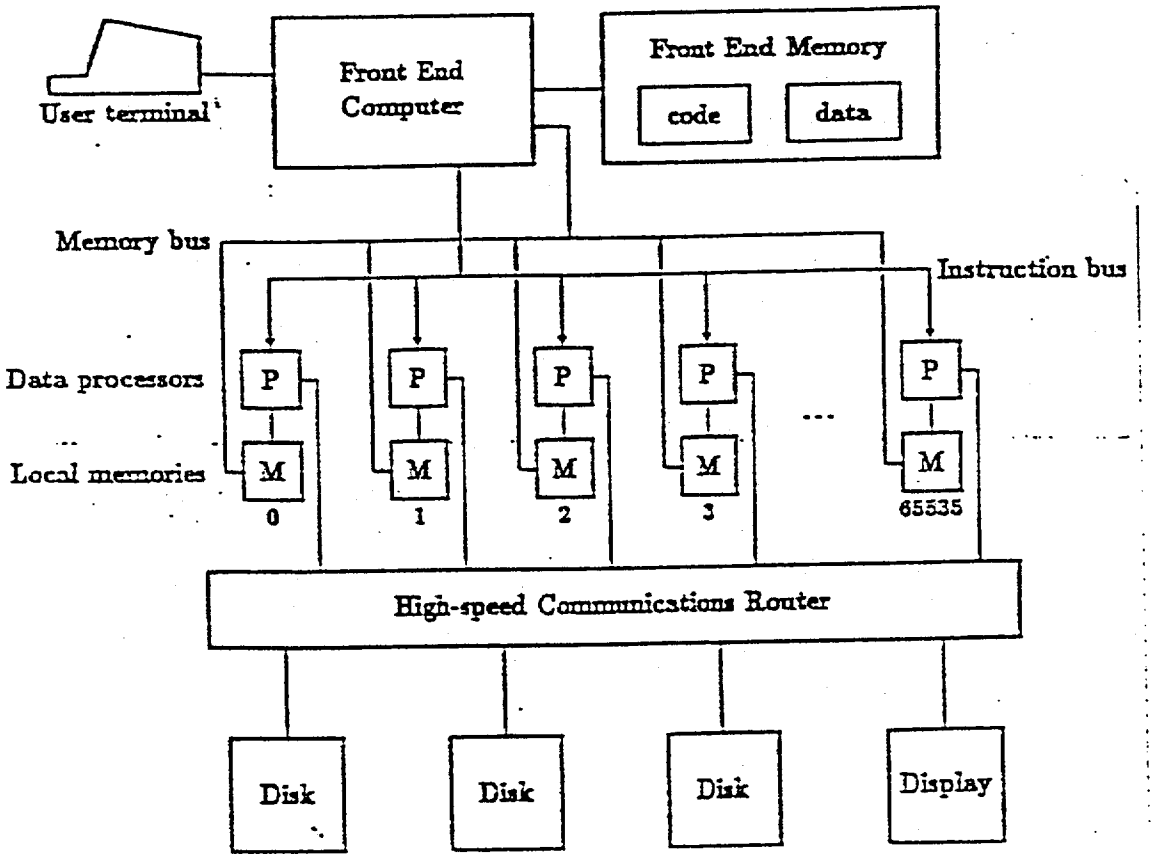

Fig. 6 Architecture of the Connection Machine. (from Thinking Machines Corporation Documentation of the Connection Machine)

Ratio n	Virtual processors	Memory each (CM-2)
1	64K	8K bytes
2	128K	4K bytes
4	256K	2K bytes
8	512K	1K bytes
16	1M	512 bytes
32	2M	256 bytes
64	4M	128 bytes
128	8M	64 bytes
256	16M	32 bytes
512	32M	16 bytes
1K	64M	8 bytes
2K	128M	4 bytes
4K	256M	2 bytes

Fig. 7 Various configurations of virtual processors and memory on the Connection Machine. (from Thinking Machines Corporation Documentation of the Connection Machine)

```
#define  NX   256   /* number of grid points in x-dir */
#define  NY   256   /* number of grid points in y-dir */
#define  NZ    32   /* number of layers */

domain state {
        float u [ NZ ];     /* u wind                    */
        float v [ NZ ];     /* v wind                    */
        float t [ NZ ];     /* potential temperature     */
        float q [ NZ ];     /* specific humidity         */
        float pstar    ;    /* surface pressure          */
        float zstar    ;}; /* terrain height            */

domain state points [ NX * NY ];
```

The u and v wind components are staggered one half grid distance in both the x and y directions from the potential temperature points on the "B" grid. In terms of the processor where they are stored, the wind components are colocated with the mass point to the "northwest".

4.2 Finite difference and averaging operators

Interprocessor communication is slow compared to the floating point performance for data within a processor. MACROS to retrieve data from neighboring processors have been coded in the Parallel Instruction Set (PARIS) of the Connection Machine using the NEWS communication. These MACROS are called XP1, XM1, YP1, and YM1 for x plus 1, x minus 1, etc. Using these MACROS, the basic horizontal coupling operators can be efficiently computed.

The C* code to compute the x-derivative, y-derivative and four point average of u wind at the mass points in parallel for some layer k is presented below.

```
temp1 = YM1 ( u [ k ] );
temp2 = temp1 + u [ k ];
temp3 = XM1 ( temp2 );

/* u bar */
ubar = ( temp2 + temp3 ) * 0.25;

/* dx of u */
dxu = ( temp2 - temp3 ) * 0.5;

/* dy of u */
temp3 = temp1 - u [ k];
dyu = ( temp3 + XM1 ( temp3 ) ) * 0.5;
```

The orientation of the points relative to each other for the above code segment is shown in Fig. 8.

It can be shown that this code segment minimizes the amount of interprocessor communication. Similar code exists to compute the other horizontal coupling terms. The update of the prognostic variables, once all the terms are computed, will occur in the processor memories at the nominal performance of the machine.

It is interesting to note that approximately 40% of the running time of the model is spent doing interprocessor communication. In other words, if data could be accesses from adjacent processors at the same speed as data within a processor the code would run 40% faster.

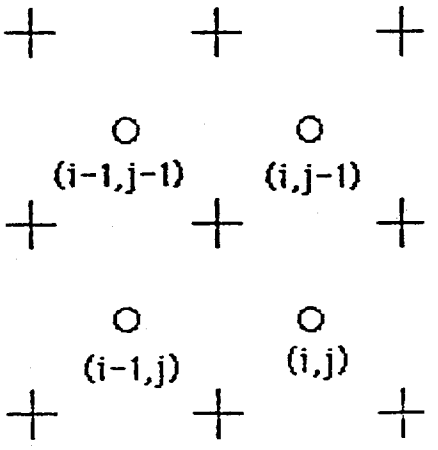

Fig. 8 Orientation of points for finite difference and averaging operators. I is the index in the x-direction and J is the index in the y-direction.

5. PERFORMANCE FIGURES

Comparing the performance of algorithms on several different computer systems can be a difficult because the organization of the code may favor one system over the other. The requirement of contiguous long vectors for "good" vector performance on the CDC CYBER 205 is a well known example of this problem. For this study, every attempt was made to be fair to all systems. In this section, the performance of the Connection Machine and several other computer systems will be presented.

5.1 Model performance on different architectures

An ANSI 77 FORTRAN version of the model was designed which should execute efficiently on most systems. The code was organized so that vectorizing compilers for vector architectures would see a vector length equal to the horizontal domain. The loops over the horizontal domain, however, contains many instructions so that non-vector architectures will see enough computational work to allow for "instruction scheduling". The CYBER 205 compiler was able to vectorize every horizontal loop. Chaining or linked triads were encouraged through the liberal use of parenthesis to

help the compilers identify opportunities for these time saving instructions.

All timings of the FORTRAN version of the code were done for a 50 x 50 horizontal grid with 32 vertical layers. The minimum grid distance on the image plane was set at 40 kms and the appropriate time step for computational stability was used. The forecast length was set for 24 hours or one forecast day. This configuration represents a limited area domain of about 2000 kms on a side. This domain size is unrealistically small and was chosen to so that CPU timings from a variety of systems with much different performance characteristics could be obtained. A realistic configuration would be a 256 x 256 horizontal grid corresponding to a domain of approximately 10000 km on a side. Since the amount of computational work is linear with the number of grid points, the timings obtained with the 50 x 50 grid can be scaled to arrive at timings for a 256 x 256 grid. For the most restrictive architecture, the CYBER 205, this is valid because a vector length of 2500 (50 X 50) is long enough for vector efficiency of over 90%. I am assuming that sufficient memory is available to hold the larger domain and that memory conflicts are not significantly changed. Assuming 64-bit floating point precision, this problem will need about 80 Megabytes of memory for a 256 x 256 grid with 32 layers. The introduction of physical parameterization for turbulence, radiation, and precipitation will increase the memory requirements.

Table 1 shows the CPU timings for various systems. The NAS 9050 is an IBM 370 plug compatible system featuring scalar processing with a 38ns clock. It performance is comparable to the CDC 205 using the scalar processor only. The Very Long Instruction Word Multiflow system does very well compared to the NAS and 205 when you consider that its price tag in about $500K. The 205, using its vector processor, achieves a speedup of about 10 to 1 for 64 bit and 20 to 1 for 32 bit over the scalar processor. These are typical values for very well vectorized code. The speed difference between the CRAY X-MP and Y-MP reflects, almost exactly, the difference in the

cycle time (8.5ns for the X-MP vs.6.2ns for an early YMP).

SYSTEM	PRECISION	COMPILER	CPUTIME
NAS 9050	32 bit	IBM VS-FORTRAN	55417 s
CDC 205(scalar)	32 bit	CDC FTN200	46784 s
CDC 205(scalar)	64 bit	CDC FTN200	40186 s
Multiflow 14/200	32 bit	TRACE FORTRAN	34524 s
CDC 205(vector)	64 bit	CDC FTN200	3382 s
CRAY X-MP(1 proc)	64 bit	CFT77	2439 s
CDC 205(vector)	32 bit	CDC FTN200	1861 s
CRAY Y-MP(1 proc)	64 bit	CFT77	1730 s

TABLE 1. CPU times for the FORTRAN version of the model. All times were computed for a 50 x 50 horizontal grid and scaled to a 256 x 256 grid. The forecast length is 24 hours, there are 32 vertical layers and the grid distance is 40 kms.

5.2 Model performance on the Connection Machine

CPU timings for the C* version of the model on the Connection Machine are presented in table 2. The domain size considered is again 256 x 256 with 32 layers. The numbers presented are a combination of measured and computed results using the 16K Connection Machine at NRL. Since this problem is completely parallel and the Connection Machine scales linearly, the computed results are accurate.

The best wall time performance is achieved when one processor is assigned to each vertical column; a VP ratio of 1. A 64K processor machine is required and it will solve the problem in a little less than half of the time required by the CRAY Y-MP using one processor. The best use of the processors in terms of speed per physical processor is obtained when several virtual processors are assigned to each physical processor. Pipelining of instructions across several virtual processors within a physical processor results in a better utilization of the hardware as shown by the CPU/VP ratios in table 2. A VP ratio of 4 is probably best because anything higher results in too little memory per processor to solve the problem once physical parameterizations are added. For a 64K machine, a 512 X 512 grid would yield a VP ratio of 4 and would take 2438 secs to solve.

PHYSICAL PROCESSORS	VP RATIO	CPUTIME	CPU/VP RATIO
64K	1	768 s	768 s
16K	4	2438 s	610 s
8K	8	4656 s	582 s

TABLE 2. CPU times for the C* version of the model with a 256 x 256 grid, 40km grid spacing, 32 layers and a forecast length of 24 hours for different numbers of virtual processors per physical processors. The CPU time normalized by the VP ratio is also presented.

5.3 Multitasking on the CRAY Y-MP

The FORTRAN version of the model was also run on the CRAY Y-MP using multiple processors. The code was multitasked using the recently available autotasking software. This software features a preprocessing step which analyzes the data dependencies within the code and inserts microtasking compiler directives into the source prior to the actual

compilation process. The processed source is available for inspection and may be further modified prior to compilation. This preprocessing step automates what was previously a manual task. As with all source preprocessors, one should expect to go through several iterations before obtaining an optimized version of the code.

The source output from the autotasking software was modified by including additional compiler directives. The FORTRAN statements remained unchanged. The timing results on the CRAY Y-MP using 4 and 8 processors is shown in Table 3. The results, indicated in terms of speedup over a uniprocessed run, indicate a point of diminishing returns with 4 processors.

NUMBER OF PROCESSORS	SPEEDUP OVER 1 PROCESSOR
4	2.6
8	2.8

TABLE 3. Speedup over 1 processor for multitasked versions of the code on the CRAY Y-MP.

The explanation of why an additional four processors failed to speedup the code significantly is as follows. Refering back to Fig. 5, the first 'k' loop contains a recursive computation for the vertical velocity and geopotential. This dependency prevents the 'k' loop from being microtasked. The microtasking is then applied to the NX x NY dimension. For the NX = NY = 50 grid dimension used in the run, the synchonization of processors at the end of each horizontal loop becomes a bottleneck. In other words, the calculations are too fine grained. The second 'k' loop contains no vertical dependencies and was microtasked. Since a majority of the work is contained in the first 'k' loop, a point was reached where additional processors could not be effectively used.

The solution, fortunately, is straight forward. An increase in the horizontal domain to the desired size of 256 x 256 should result in significantly less synchonization overhead at the end of each horizontal loop. Alternatively, a seperate 'k' loop could be constructed to compute the vertically coupled portions of the inertia-gravity wave calculations. The first 'k' loop would not contain any vertical dependencies and could then be microtasked. This additional 'k' loop would obviously microtask over the horizontal loop but would result in less code being subjected to a fine-grained bottleneck. A significant increase in temporary storage, however, would be required. Unfortunately, as of this time neither of these opportunities has been pursued.

6. SUMMARY AND CONCLUSION

Solution of the primitive equations on the CM has been found to be straight forward and very efficient. The execution time for the dynamics is comparable to vector supercomputers. This paper did not consider physical parameterizations, however, some general conclusions can be reached. Since a processor was assigned to each column and physical parameterizations generally do not involve horizontal information exchange, it can be anticipated that physics can be computed very quickly on the CM. The nominal floating point performance of the machine, 1 Gigaflop, should be achievable as all calculations will involve data already in the processor memories. Perhaps more importantly, physical parameterization routines can be coded as serial code with looping over 'k'; a very natural way of thinking. The projection of the code so as to execute on all processors simultaneously is straighforward.

7. ACKNOWLEDGEMENTS

I would like to thank John Church of NRL for helping me get started on the Connection Machine and for being available to answer many questions. Robert Whaley of Thinking Machines

Corporation wrote the MACROS used for interprocessor communications and offered many useful suggestions. Jim Abeles, Nic Talian and Steve Perry of Cray Research generously volunteered to run the model on the Cray systems and discuss the results with me. Chuck Aston and Louis Hackerman of Multiflow Computer also volunteered to run the model on their system and supply me with the results. I would like to especially thank Fran Balint for giving me the time necessary to investigate the Connection Machine.

8. REFERENCES

Arakawa, A., 1972: Design of the UCLA general circulation model, Numerical Simulation of Weather and Climate, Dept. of Meteorology, Univ. of California, Los Angeles, Tech. Rept. No. 7.

Gadd, A.J., 1974: An economical explicit integration scheme. Meteor. Office Tech. Note 44, 7 pp.

Hillis, D.W., 1986: The Connection Machine. MIT Press, Cambridge, 1986.

Lax, P., and B. Wendroff, 1960: Systems of conservation laws. Comm. Pure and Appl. Math., 13, 217-237.

Phillips, N.A., 1957: A coordinate system having some special advantages for numerical forecasting. J. Meteor., 14, 184-185.